Partiel de la matière Web Services

Consignes pour le passage du partiel : L'examen se fera sur une durée de 2 h 30. Il sera à faire sur l'ordinateur portable de l'étudiant. Les documents numériques ou papiers seront autorisés. L'accès au réseau et à interne est refusé.

Consignes pour le rendu du partiel : Le partiel devra être rendu sur une clef USB que les surveillants vous transmettront à la fin de l'examen. Vous devrez créer un dossier formaté de la façon suivante « *VOTRE_NOM votre_prénom* ». C'est dans ce dossier que vous devrez stocker le rendu de votre partiel. Les réponses aux QCM de l'exercice n°1 devront être stockées dans un fichier texte nommé « *Exercice1.txt* ». Pour les exercices n°2 et n°3, les projets eclipse devront être transmis.

Exercice n°1 : QCM - 5pts

1. **Un consommateur développé en Java peut-il communiquer avec un Web Service développé en C#.net ?**
   A – oui B – non
2. **Un consommateur peut-il connaitre le fonctionnement interne d'un Web Service ?**
   A – oui B – non
3. **Un Web Service REST est :**
   A – un standard B – un style d'architecture
4. **Un Web Service REST s'appuie sur le protocole :**
   A – FTP B – SMTP C – HTTP
5. **Sur le développement d'un Web Service REST, quelle annotation permet de spécifier le type de données (XML, JSON, TEXT, etc.) que va renvoyer une méthode ?**
   A - @Context B - @Consumes C - @Produces

Exercice n°2 : Développement d'un Web Service étendu côté serveur – 9 pts

L'objectif de cet exercice est de développer une calculatrice basique sous la forme d'un Web Service. Elle ne devra faire que des opérations simples (addition, soustraction, multiplication, division) entre seulement deux opérandes. Chaque opérateur de la calculatrice aura une opération du Web Service qui lui est associé.

Ce Web Service devra être développé en suivant l'approche **Bottom/Up**. Les noms des méthodes java représentant les opérations du web service sont les suivantes :

- **additionner**
- **soustraire**
- **multiplier**
- **diviser**

Les deux opérandes qui doivent être passées en paramètres seront de type **float**. Le résultat retourné par la méthode sera aussi de type **float**.

Le nom de votre Web Service sera : **Calculatrice**.

Le nom du portType de votre Web Service sera : **CalculatricePort**.

Les noms des opérations (requête/réponse) de votre Web Service associées à vos méthodes java seront :

- **additionnerRequete**
  additionnerResultat
- **soustraireRequete**
  soustraireResultat
- **multiplierRequete**
  multiplierResultat
- **diviserRequete**
  diviserResultat

Les noms des deux opérandes qui seront passées en paramètres à la requête seront :

- **operande1**
- **operande2**

Votre Web Service devra être stocké dans un projet web nommé :
PartielCalculatriceWebServiceExercice2

L'adresse URL pour accéder à votre web service devra être de la forme :

http://localhost:8083/PartielCalculatriceWebServiceExercice2/calculatrice

En testant votre Web Service dans le logiciel SOAP-UI, vous devriez avoir cela en enveloppe SOAP pour votre requête :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cal="ht
   <soapenv:Header/>
   <soapenv:Body>
      <cal:additionnerRequete>
         <operandel>1</operandel>
         <operande2>2</operande2>
      </cal:additionnerRequete>
   </soapenv:Body>
</soapenv:Envelope>
```

Et cela en enveloppe SOAP pour votre réponse :

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:additionnerRequeteResponse xmlns:ns2="http://calculatricewebserviceexercice
         <additionnerResultat>3.0</additionnerResultat>
      </ns2:additionnerRequeteResponse>
   </S:Body>
</S:Envelope>
```

Le fonctionnement du Web Service ne donnera pas la totalité des points. Le respect des consignes notamment sur le nommage des opérations, des opérandes, etc. est aussi noté.

Exercice n°3 : Développement d'un Web Service étendu côté client – 6 pts

L'objectif de cet exercice et de développer un consommateur sous la forme d'une application Web qui communiquera avec le Web Service développé précédemment. Ce consommateur proposera une interface graphique pour la saisie des deux opérandes et la sélection de l'opérateur. Dans cette même interface, nous afficherons le résultat de l'opération.

Le développement du consommateur devra être fait-en suivant l'approche Top/Down.

Votre consommateur devra être stocké dans un projet web nommé :
PartielCalculatriceWebServiceClientExercice3

Afin de vous faire gagner du temps, vous trouverez en annexe, les sources du fichier « *index.jsp* », qui gère toute la partie interface graphique et qui ne nécessite aucune modification. Vous trouverez aussi en annexe les sources de la servlet, qui va extraire les informations retournées par le formulaire de la JSP « *index.jsp* ». Des modifications seront nécessaires dans la servlet pour faire appel au Web

Service et retourner à la JSP « *index.jsp* » le résultat que lui aura retourné le Web Service (pour plus d'informations sur les traitements attendus, se référer au commentaire TODO présent dans les sources de la servlet).

Voici les résultats attendus :

Je ne veux voir aucun traitement de calcul dans le code source du consommateur. Vous devez obligatoirement passer par le Web Service pour réaliser les opérations. Si vous ne respectez pas cette consigne, cela entrainera un 0 sur l'exercice.

Annexe n°1 : index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
      <h1>Consommateur du Web Service Calculatrice</h1>
      <form method="post" action="Calculer">
            <input type="text" name="operande1" value="<%= request.getParameter("operande1")%>"/>
            <select name="operateur">
                  <option>+</option>
                  <option>-</option>
                  <option>*</option>
                  <option>/</option>
            </select>
            <input type="text" name="operande2" value="<%= request.getParameter("operande2")%>"/>
            <input type="submit" value="="/>
            <%if(request.getAttribute("resultat")!=null && !request.getAttribute("resultat").equals("")){ %>
                  <%=request.getAttribute("resultat") %>
            <%}else { %>
                  ...
            <%} %>
      </form>
</body>
</html>
```

Annexe n°2 : CalculerServlet.java

```
package servlets;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class CalculerServlet
 */
public class CalculerServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
	 *      response)
	 */
	protected void doGet(HttpServletRequest request,
			HttpServletResponse response) throws ServletException, IOException {
		String operande1Str = request.getParameter("operande1");
		String operateurStr = request.getParameter("operateur");
		String operande2Str = request.getParameter("operande2");
		String resultatARetourner = "";
		// TODO ajouter les traitements adéquats pour traiter les informations
		// renvoyées par le formulaire
		// de la JSP index.jsp en faisant appel au Web Service Calculatrice.
		// Il faut penser à vérifier avant de faire appel au web service, si les
		// opérandes sont non nulles ou bien ne sont
		// pas des chaines de caractères vides ou encore si ce sont bien des
		// nombres. Si une de ces conditions,
		// n'est pas valide, alors afficher un message d'erreur à la place du
		// résultat informant l'utilisateur de la cause
		// de l'erreur.
		request.setAttribute("resultat", resultatARetourner);
```

```
            request.getRequestDispatcher("index.jsp").forward(request, response);
      }

      /**
       * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
       *      response)
       */
      protected void doPost(HttpServletRequest request,
                  HttpServletResponse response) throws ServletException, IOException {
            doGet(request, response);
      }
}
```

Code à intégrer dans votre fichier « *web.xml* » pour le fonctionnement de la servlet :

```
      <servlet>
            <description></description>
            <display-name>Calculer</display-name>
            <servlet-name>Calculer</servlet-name>
            <servlet-class>servlets.CalculerServlet</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>Calculer</servlet-name>
            <url-pattern>/Calculer</url-pattern>
      </servlet-mapping>
```